Data: $D = \langle (x_n, y_n) \rangle_{n=1}^{N}$, number of epochs $E$, weighted learner $\mathcal{W}$

Result: classifier

$\beta^{(0)} = \langle \frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N} \rangle$; # initialize example weights

for $e \in \{1, \ldots, E\}$ do

$f^{(e)} \leftarrow \mathcal{W}(D, \beta^{(e-1)})$; # train the classifier on the weighted data

$\hat{\epsilon}^{(e)} \leftarrow \sum_{n=1}^{N} \beta^{(e-1)}_n \cdot \mathbb{I}[f^{(e)}(x_n) \neq y_n]$; # weighted error rate

$\alpha^{(e)} \leftarrow \frac{1}{2} \log \left( \frac{1 - \hat{\epsilon}^{(e)}}{\hat{\epsilon}^{(e)}} \right)$; # “adaptive” weight for $f^{(e)}$

for $n \in \{1, \ldots, N\}$ do

$\beta^{(e)}_n \leftarrow \frac{1}{Z^{(e)}} \cdot \beta^{(e-1)}_n \cdot \exp (-\alpha^{(e)} \cdot y_n \cdot f^{(e)}(x_n))$; # update example weights

($Z^{(e)}$ is a normalization constant)

end

end

return $f_{\text{boost}}(\cdot) = \text{sign} \left( \sum_{e=1}^{E} \alpha^{(e)} \cdot f^{(e)}(\cdot) \right)$;

Algorithm 1: AdaBoost
Notes about AdaBoost

- Typically, $\mathcal{W}$ is a shallow decision tree, or a linear classifier. In the literature, it is often called a \textbf{weak} learner (definition comes later).

See the book for more insight on what happens on the first epoch with a very simple $\mathcal{W}$. Each successive $f(e)$ is intended to work harder wherever previous classifiers have been failing (hence, "adaptive").
Notes about AdaBoost

- Typically, $\mathcal{W}$ is a shallow decision tree, or a linear classifier. In the literature, it is often called a **weak** learner (definition comes later).
- $\alpha$ as a function of $\hat{e}$:

![Graph showing the relationship between error and weight]

For examples we get right ($f(e(x_n)) = y_n$), the weight $\beta_n$ will decrease; we increase the weights of examples we get wrong.

- See the book for more insight on what happens on the first epoch with a very simple $W$. Each successive $f(e)$ is intended to work harder wherever previous classifiers have been failing (hence, "adaptive").
Notes about AdaBoost

- Typically, $W$ is a shallow decision tree, or a linear classifier. In the literature, it is often called a weak learner (definition comes later).

- $\alpha$ as a function of $\hat{e}$: goes to $+\infty$ when error is low, zero as error increases to 0.5.

- See the book for more insight on what happens on the first epoch with a very simple $W$. Each successive $f(e)$ is intended to work harder wherever previous classifiers have been failing (hence, "adaptive").
Notes about AdaBoost

- Typically, $\mathcal{W}$ is a shallow decision tree, or a linear classifier. In the literature, it is often called a weak learner (definition comes later).
- $\alpha$ as a function of $\hat{e}$: goes to $+\infty$ when error is low, zero as error increases to 0.5.
- For examples we get right ($f^{(e)}(x_n) = y_n$), the weight $\beta_n$ will decrease; we increase the weights of examples we get wrong.
Notes about AdaBoost

► Typically, \( \mathcal{W} \) is a shallow decision tree, or a linear classifier. In the literature, it is often called a weak learner (definition comes later).

► \( \alpha \) as a function of \( \hat{\epsilon} \): goes to \( +\infty \) when error is low, zero as error increases to 0.5.

► For examples we get right \( (f^{(e)}(x_n) = y_n) \), the weight \( \beta_n \) will decrease; we increase the weights of examples we get wrong.
  
  ► See the book for more insight on what happens on the first epoch with a very simple \( \mathcal{W} \). Each successive \( f^{(e)} \) is intended to work harder wherever previous classifiers have been failing (hence, “adaptive”).
Boosting Example

(This is a contrived example; it may take many more iterations to achieve $\hat{\epsilon} = 0$.)
Boosting Example

(This is a contrived example; it may take many more iterations to achieve $\hat{e} = 0$.)
Boosting Example

(This is a contrived example; it may take many more iterations to achieve $\hat{\epsilon} = 0$.)
Boosting Example

(This is a contrived example; it may take many more iterations to achieve $\hat{\epsilon} = 0$.)
Boosting Example

(This is a contrived example; it may take many more iterations to achieve $\hat{\epsilon} = 0$.)

Weak Learners

Formally, a weak learner is one with $\epsilon < \frac{1}{2}$.
(These tend to be high-bias, low-variance classifiers.)
Theory says: if you can find a weak learner every round, boosting’s training error will eventually go to zero (as $E \to +\infty$).
Boosting: Make a Weak Learner Strong

Theory says: if you can find a weak learner every round, boosting’s training error will eventually go to zero (as $E \to +\infty$).

- This is non-obvious (proving it requires the use of telescoping sums):

$$\frac{1}{N} \sum_{n=1}^{N} \left[ f_{\text{boost}}(x_n) \neq y_n \right] \leq \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \cdot \sum_{e=1}^{E} \alpha^{(e)} \cdot f^{(e)}(x_n) \right)$$

"loss"
Theory says: if you can find a weak learner every round, boosting’s training error will eventually go to zero (as \( E \rightarrow +\infty \)).

▶ This is non-obvious (proving it requires the use of telescoping sums):

\[
\begin{align*}
\text{training error} & \\
\frac{1}{N} \sum_{n=1}^{N} \mathbb{1}[f_{\text{boost}}(x_n) \neq y_n] & \leq \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \cdot \sum_{e=1}^{E} \alpha^{(e)} \cdot f^{(e)}(x_n) \right)
\end{align*}
\]

▶ Our update of \( \alpha^{(e)} \) on each round is provably the choice that minimizes this loss.
Boosting: Make a Weak Learner Strong

Theory says: if you can find a weak learner every round, boosting’s training error will eventually go to zero (as $E \to +\infty$).

- This is non-obvious (proving it requires the use of telescoping sums):

$$
\frac{1}{N} \sum_{n=1}^{N} \left[ f_{\text{boost}}(x_n) \neq y_n \right] \leq \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \cdot \sum_{e=1}^{E} \alpha^{(e)} \cdot f^{(e)}(x_n) \right)
$$

- Our update of $\alpha^{(e)}$ on each round is provably the choice that minimizes this loss.
- Assuming each $\epsilon^{(e)} < \frac{1}{2}$, it’s possible to prove:

$$
\ldots \leq \exp -2 \sum_{e=1}^{E} \left( \frac{1}{2} - \hat{\epsilon}^{(e)} \right)^2
$$

(i.e., as $E$ goes up, training error decreases exponentially!)
Theory and Practice

Boosting tends to be very robust to overfitting, with out-of-sample error continuing to decrease even when training error stabilizes.

Eventually, it will overfit.

Theory gives some insight about this; PAC-style generalization bound is:

$$\epsilon \leq \hat{\epsilon} + \tilde{O}\left(\sqrt{\frac{E \cdot d}{N}}\right)$$

where $d$ measures the size of the hypothesis class.
Boosting as Loss Minimization (Exponential Loss)

The above analysis leads to another insight: boosting is minimizing yet another loss function!

Let $a(x)$ denote a score (or activation function) for input $x$—the value whose sign we take for binary classification.

$$\exp(-y \cdot a(x))$$

(Compare to log loss, $\log(1 + \exp(-y \cdot a(x)))$.)
Boosting as Loss Minimization (Exponential Loss)

The above analysis leads to another insight: boosting is minimizing yet another loss function!

Let \( a(x) \) denote a score (or activation function) for input \( x \)—the value whose sign we take for binary classification.

\[
\exp(-y \cdot a(x))
\]

(Compare to log loss, \( \log(1 + \exp(-y \cdot a(x))) \).

If \( a \) were (sub)differentiable with respect to continuous parameters, you could directly minimize exponential loss using SGD.
That’s not the case if \( \mathcal{W} \) is, say, a decision tree learner.
Palate Cleanser: Random Forests

Fix tree structure; randomly fill in features.

Do this $E$ times; let them vote.

With large enough $E$, useless trees will cancel each other out.